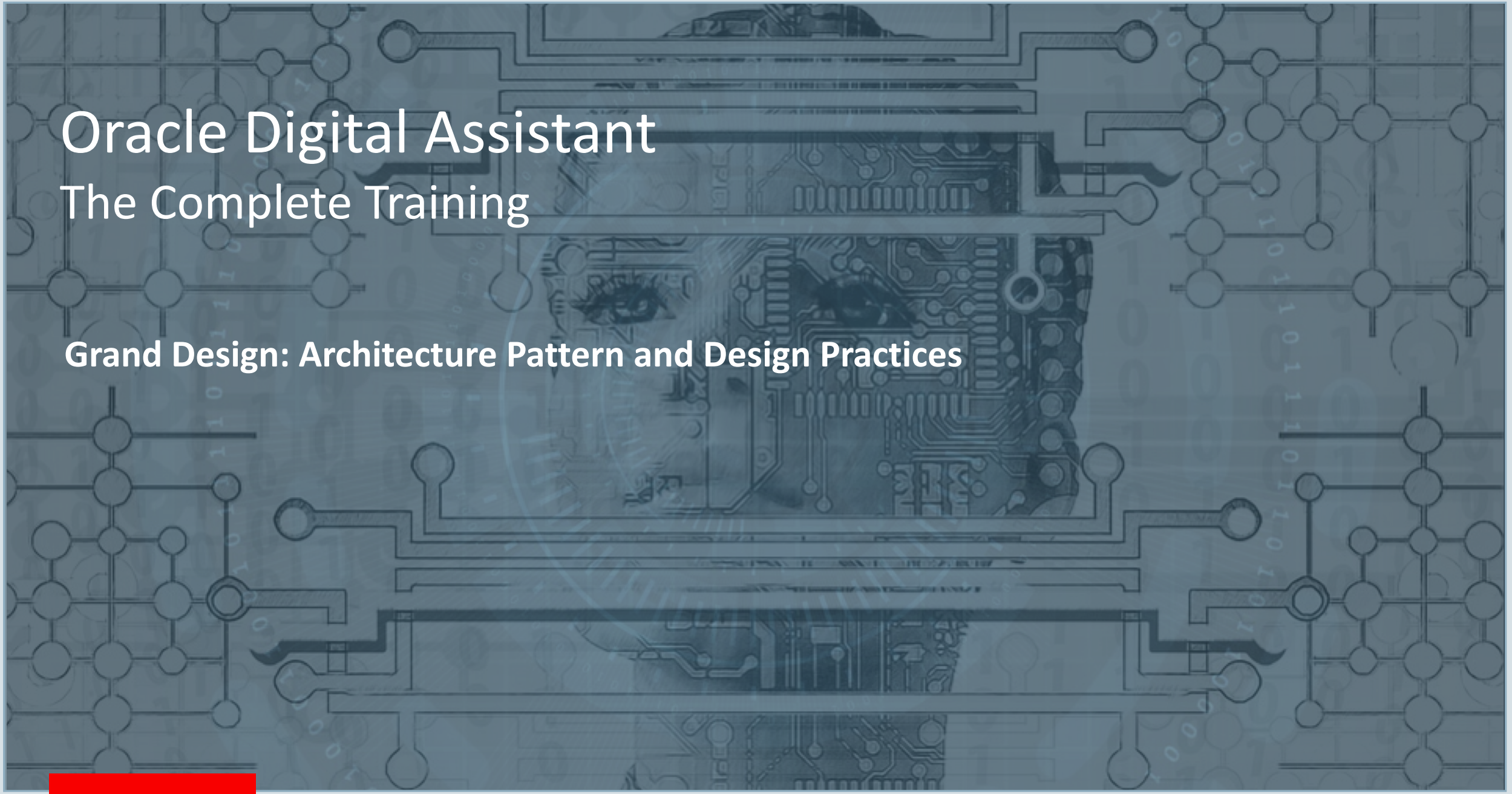

ORACLE®

# Oracle Digital Assistant

## The Complete Training

**Grand Design: Architecture Pattern and Design Practices**



ORACLE

Image courtesy of pixabay.com

# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

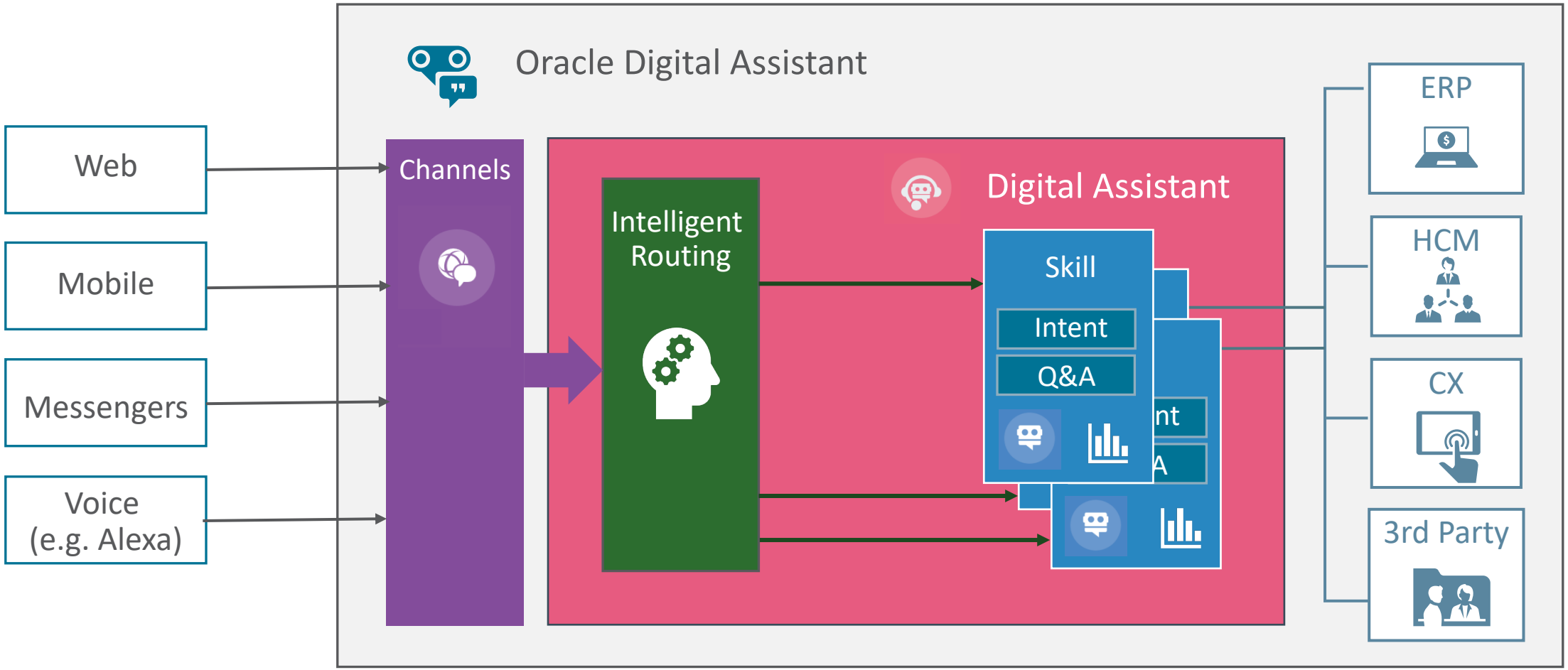
# Topic agenda

1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters

# Topic agenda

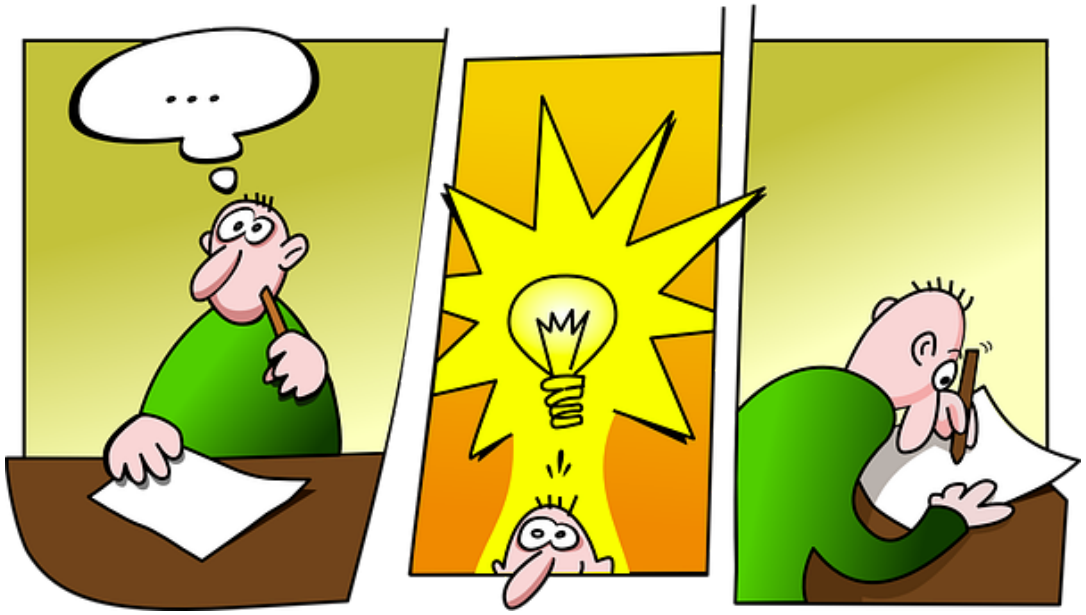
1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters

# Oracle Digital Assistant architecture



# Artificial intelligence alone doesn't build the bot

- Requires conversation design skills
- Uses a mix of conversational AI and dialog flow to assist users
- Artificial intelligence does not replace good design practices
- Building chatbots is a software development project

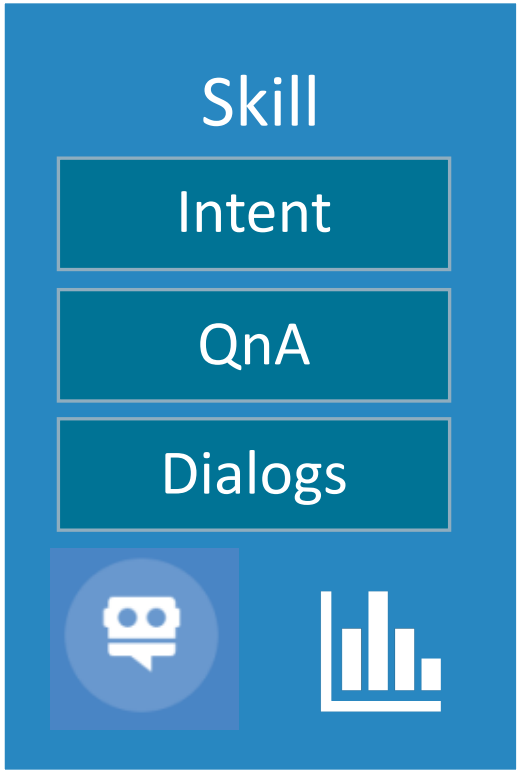


# Topic agenda

1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters

# About skills

- Skills are units of work
  - Assist users in completing a conversational task
  - Access remote services and backends
  - Do not make assumptions about the existence of user scope variables
- Scope of a skill can be
  - single use case
  - multiple use cases
  - complete business solution
- Supports modularization



An **analogy to Web development** helps to identify Oracle Digital Assistant development patterns

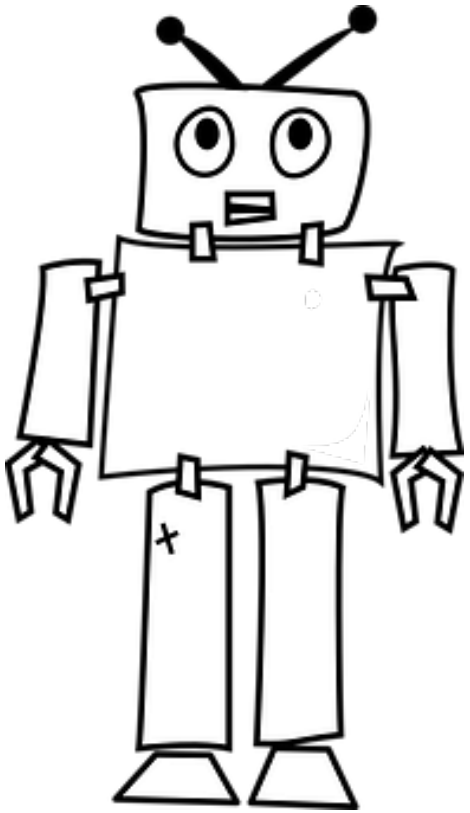
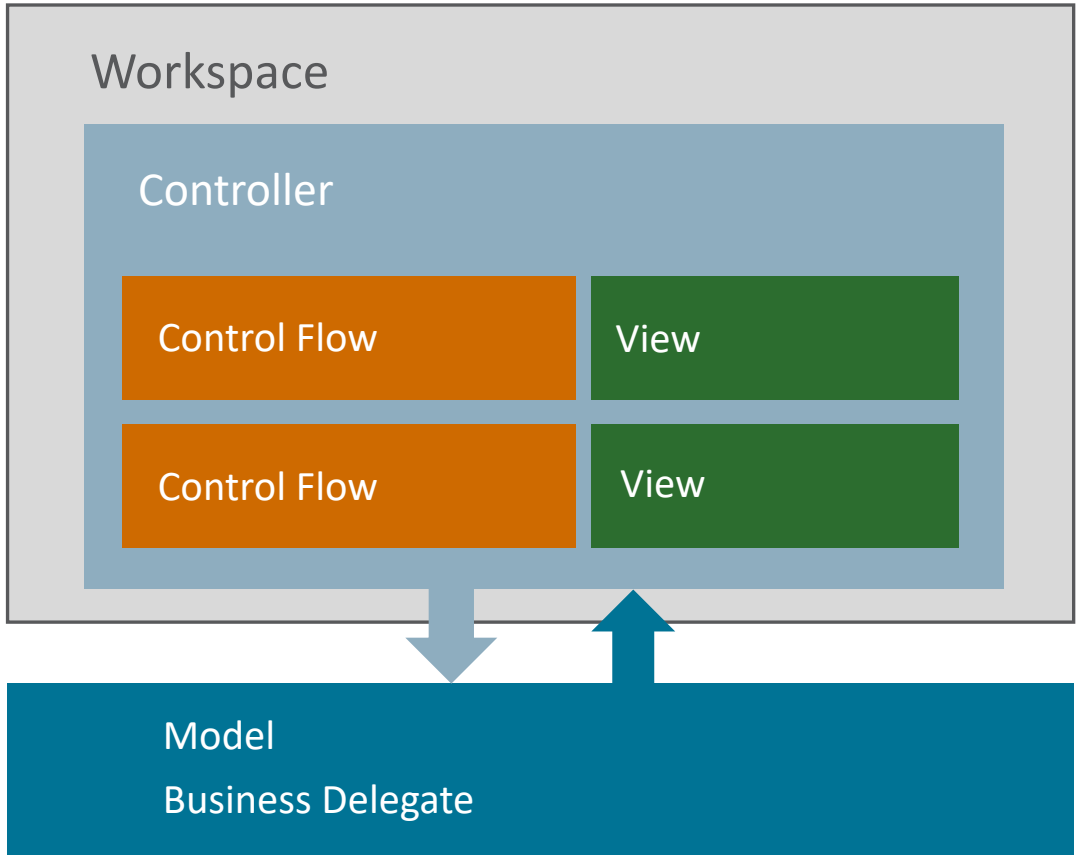


Image courtesy of pixabay.com

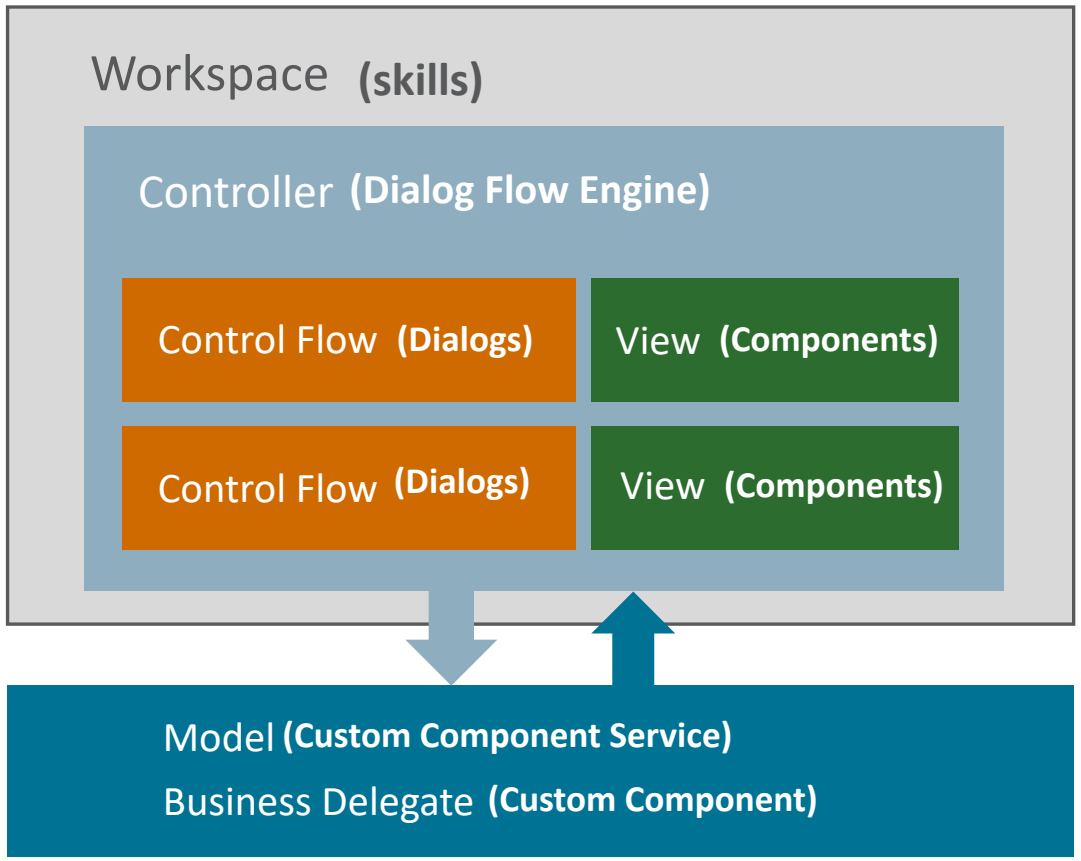
# Thinking in patterns: Web analogy

- Workspace
  - Holds project code, libraries
- Controller
  - Navigates UI and holds state
- View
  - Renders application response
- Model
  - Data and business service access
- Business delegate



# Thinking in patterns: Oracle Digital Assistant

- Skill (Workspace)
  - Holds conversations, intents, utterances and custom logic
- Dialog Flow Engine (Controller)
  - Navigates between dialogs, holds state
- Components (View)
  - Renders bot responses
- Custom Component Service (Model)
  - Data and business service access
- Custom component (Business delegate)



# Architectures

## All-in-one

- Skill as complete business solution
  - Finance, Pizza, Retail
- Skills as stand-alone solutions
- Reusability not a primary concern
- Skills are most likely built by different teams
  - Risk of inconsistent behavior and look

## Part-of-a-whole

- Skill as part of a whole
  - Member registration, course booking, message board, meeting organizer
- Small skills built with reuse in mind
- Very likely built by same team
  - Easier to enforce design principles and guidelines for a consistent appearance

# Design Patterns

## Custom Components

You build custom components for **backend integration** and to implement custom logic.

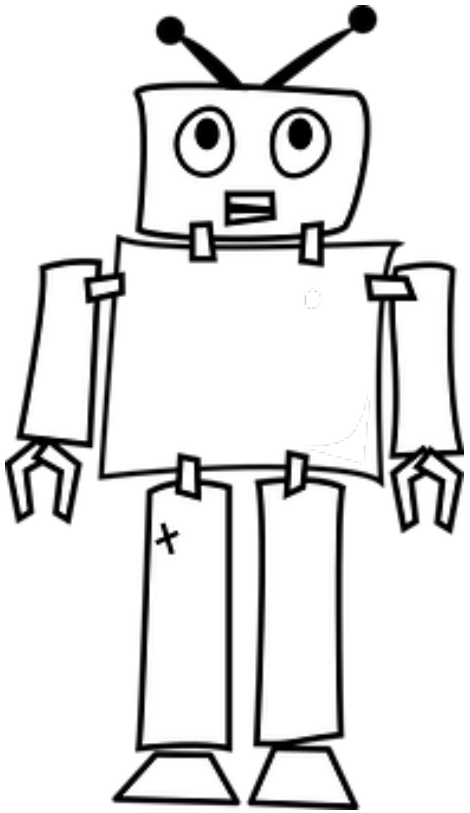


Image courtesy of pixabay.com

# Custom component deployment considerations

## Exclusive Component Service

### Local Component Container

- Skills can run different versions of a component
- Breaking a component in an update does not impact all skills but only those updated to the new version
- Error correction requires updating all deployments

- No credential store
- Component code exported with skill

## Shared Component Services

### Remote Node Servers

- Component source code resides on server
- Option to share code with other applications
- Remote server may act as a data and service integration layer
- Single point of development and maintenance

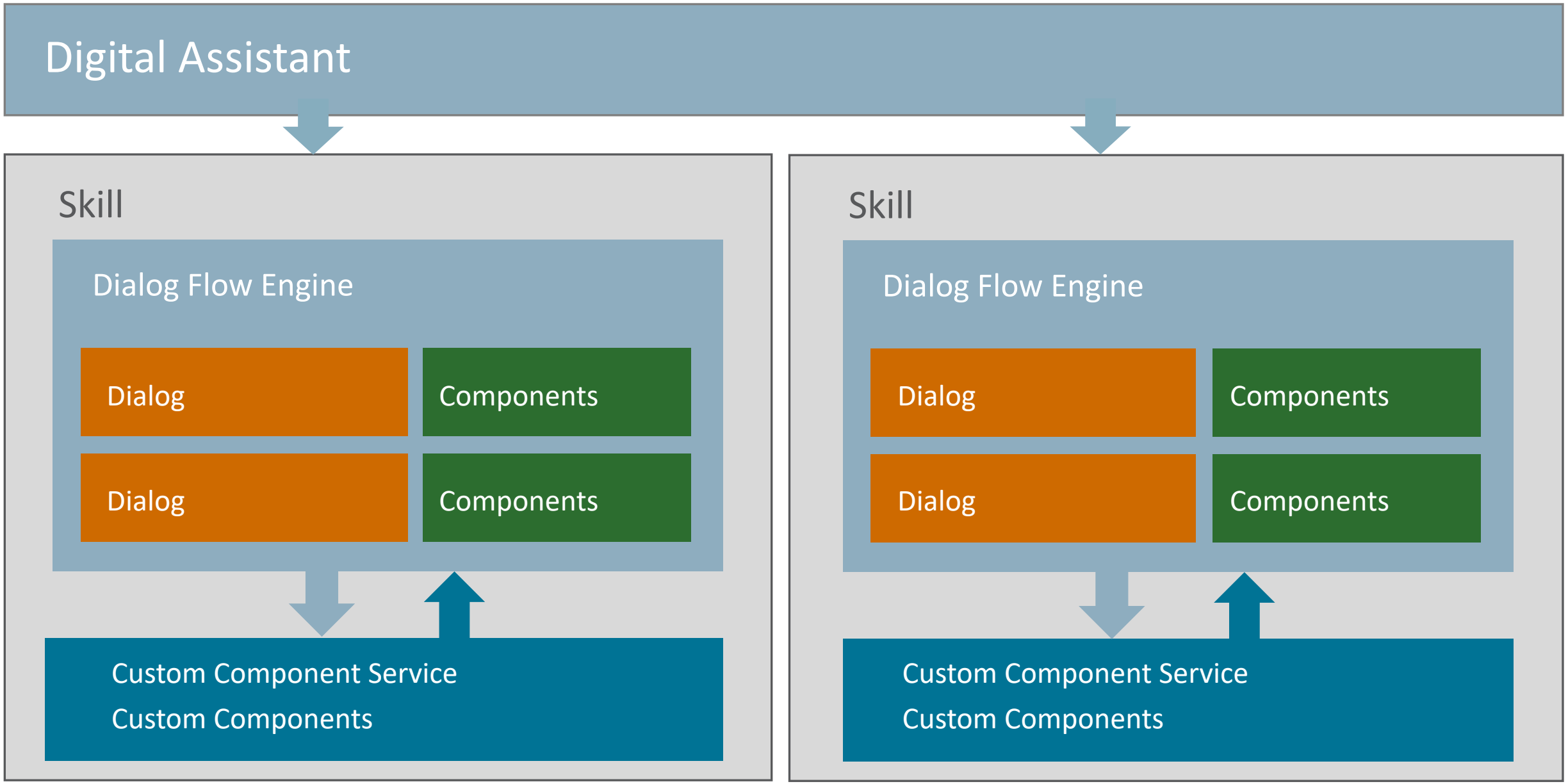
- Failure may impacts multiple skills

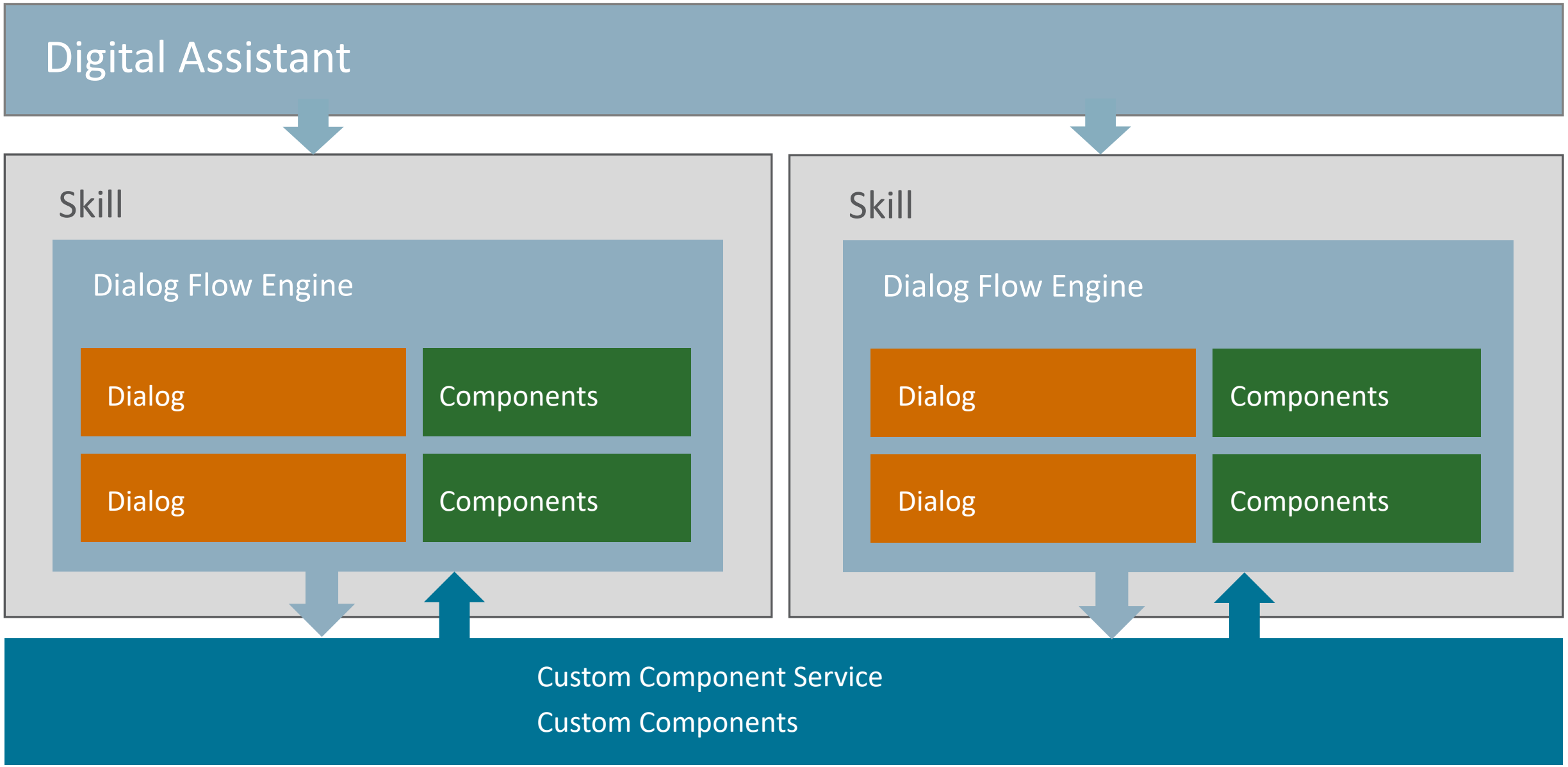
### Mobile Hub

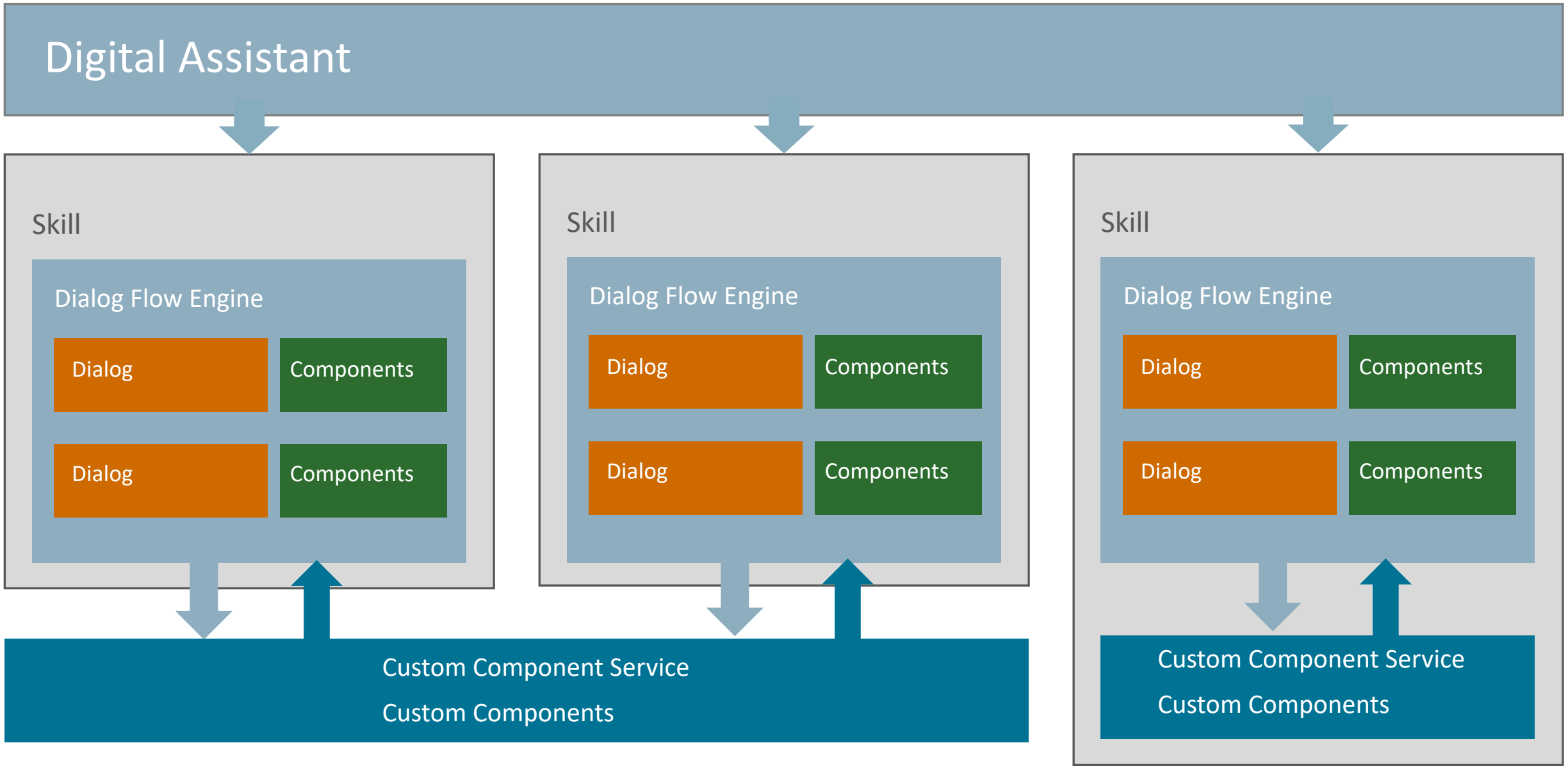
- Multi channel backend service
- Provides platform services and declarative service connectors
- Secure container
- Storage options

### 3rd Party Node Servers

- Allow use of environment variables (configuration)







# Design Patterns

## Referencing Skills From Skills

The System.Intent component accesses the skill's **NLP model** to resolve the user intent and to extract entities from the user message

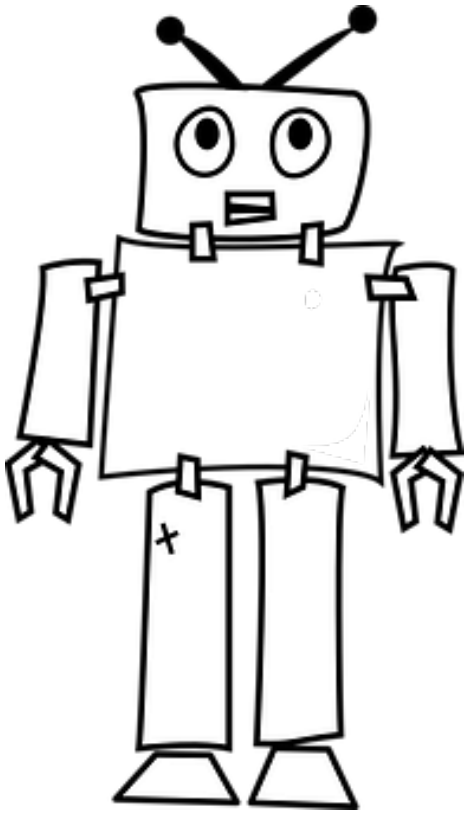
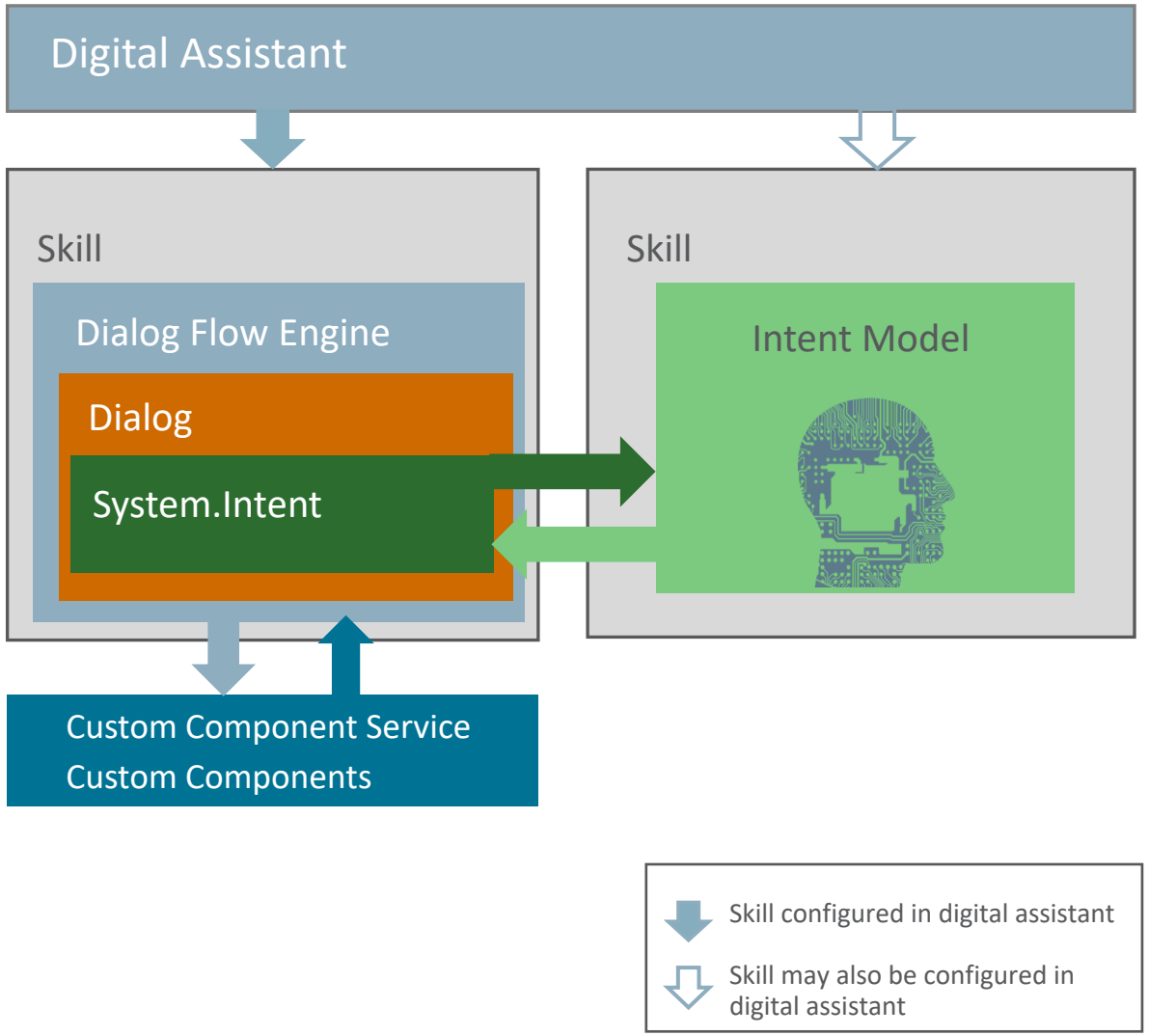


Image courtesy of pixabay.com

# Interskill referencing

- A skill's `System.Intent` state may reference another skill's intent model
- Allows one skill to share another skill's intents
- Resolved intent name and entities are returned to calling skill
- Use cases
  - Common intents (train once)
  - Use of different intent engine



Inter-Skill referencing does not execute the referenced skill's dialog flow. All intent handling happens in the skill that references the intent model of another skill.

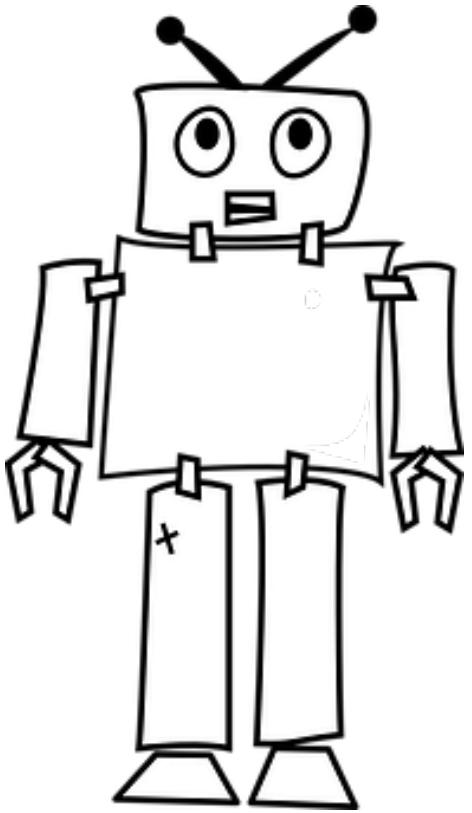
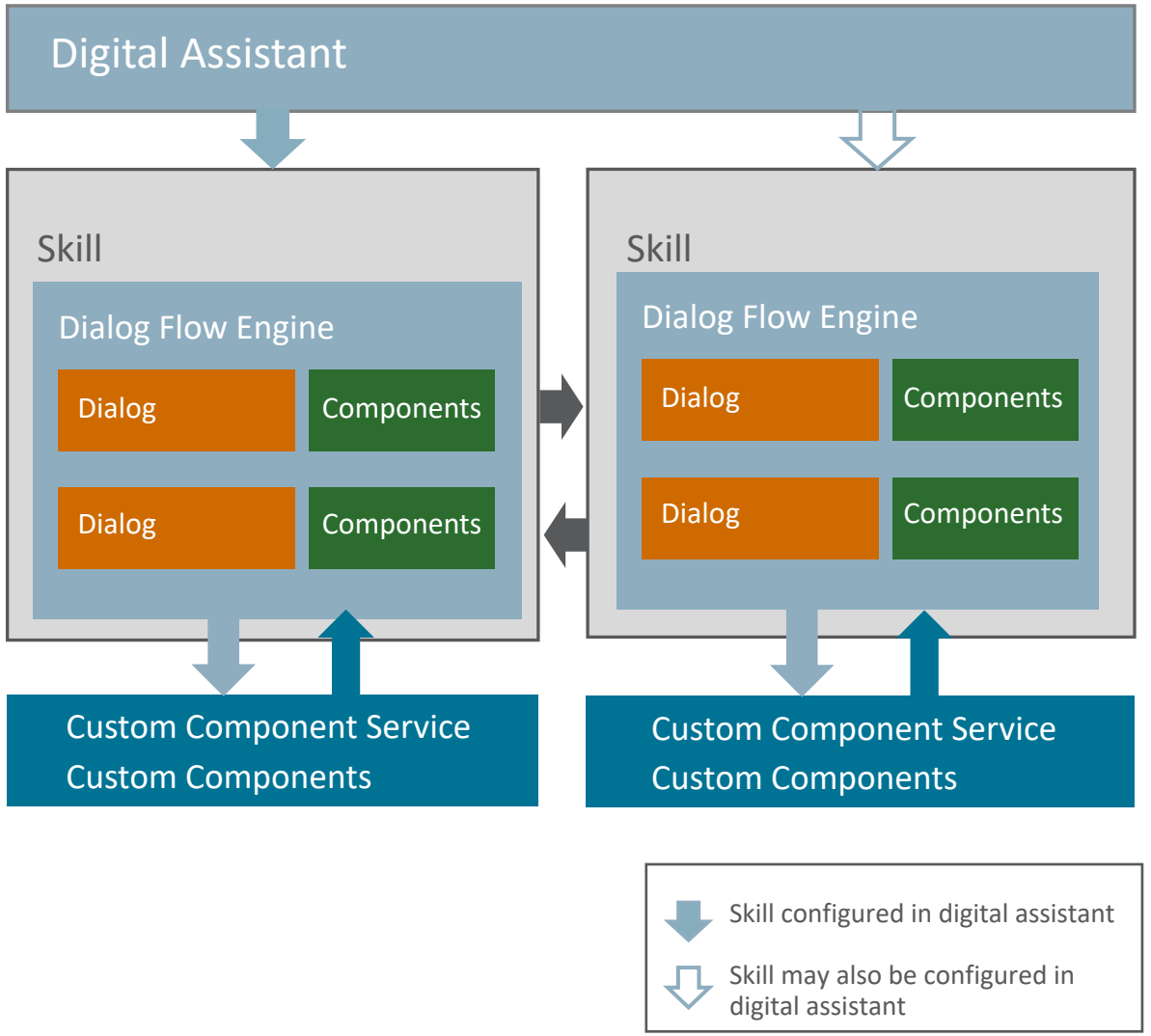
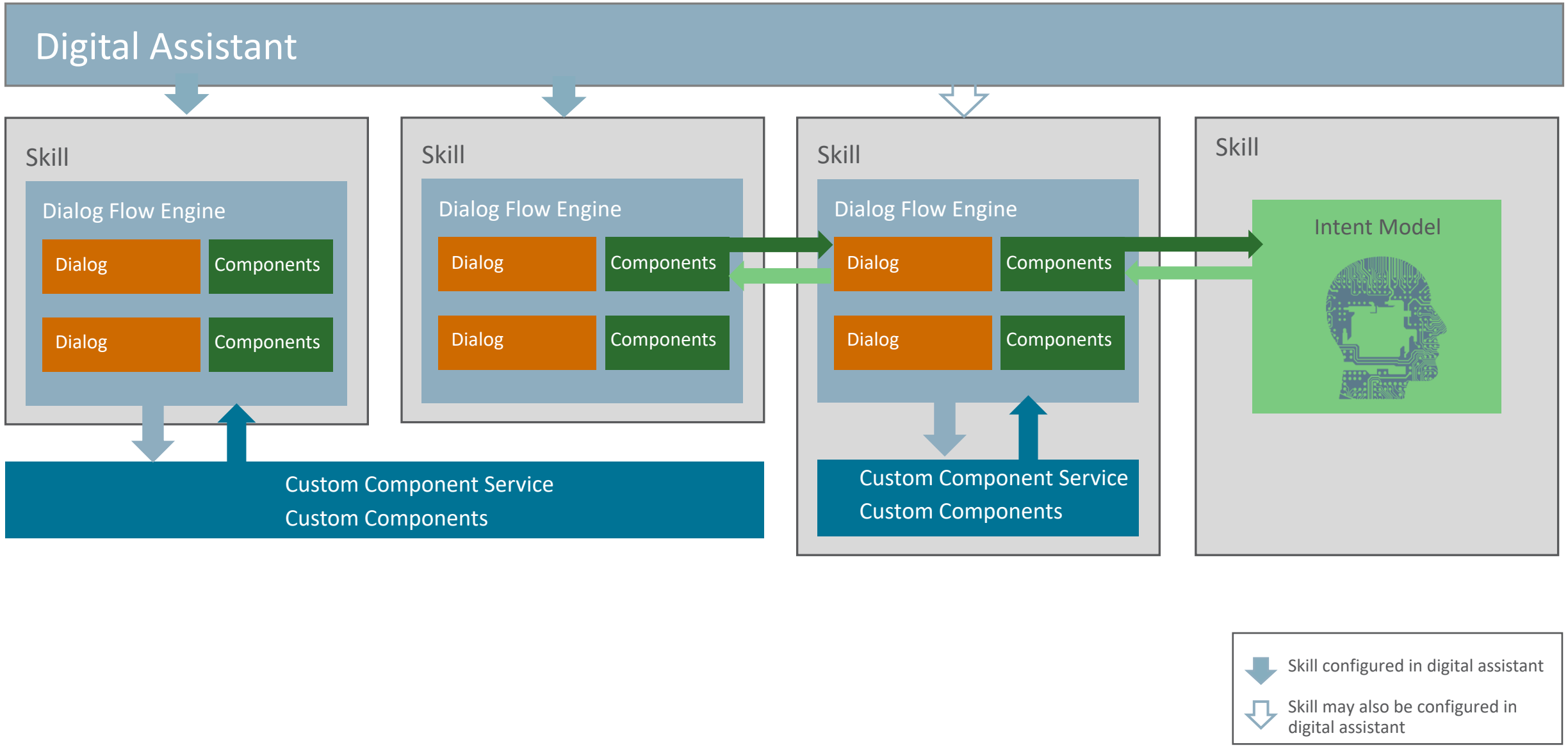


Image courtesy of pixabay.com

# Skill calling skill

- Calling skill
  - Uses `System.CommonResponse` component
  - Passes message to digital assistant





# Topic agenda

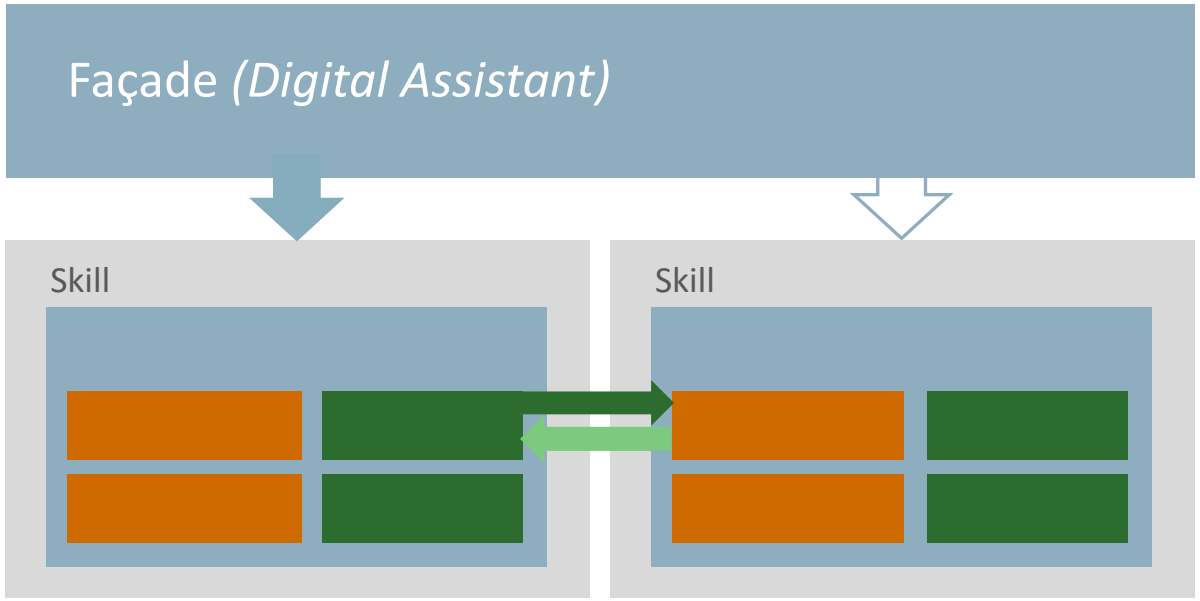
1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters

# About digital assistant

- Front-end bot that redirects user messages to one of its configured skills
  - Routing is based on intents and utterances, context and direct addressing
- Disambiguates user messages if required
- Exposed on one or many messenger channels
  - Messenger payload differences handled by configured channel connector
- Design time for digital assistant designers
  - Create chatbots by orchestrating individual skills
    - Configuration only (no coding)
  - Digital assistant designers may or may not be the skill developer

# Thinking in patterns: Digital Assistant

- Façade
  - Single entry point for a bot
  - Skills "hidden" from user view
  - Skills can be added, removed or updated without re-deployment



**Define a clear goal for your digital assistant** that is different from "all you can eat".

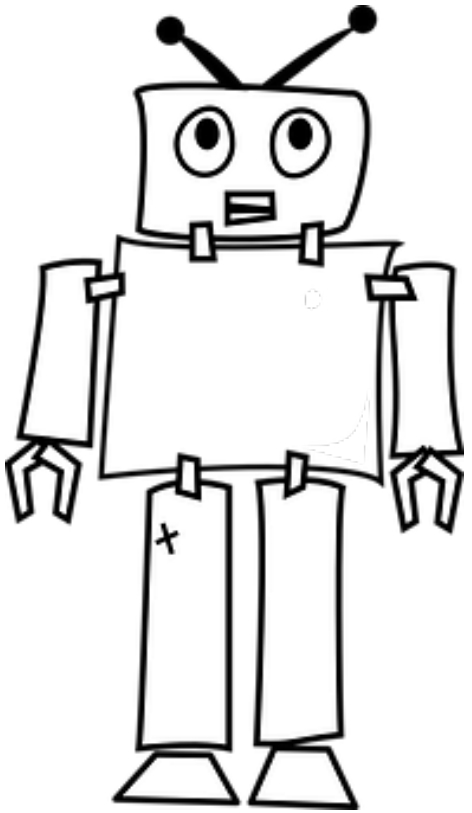
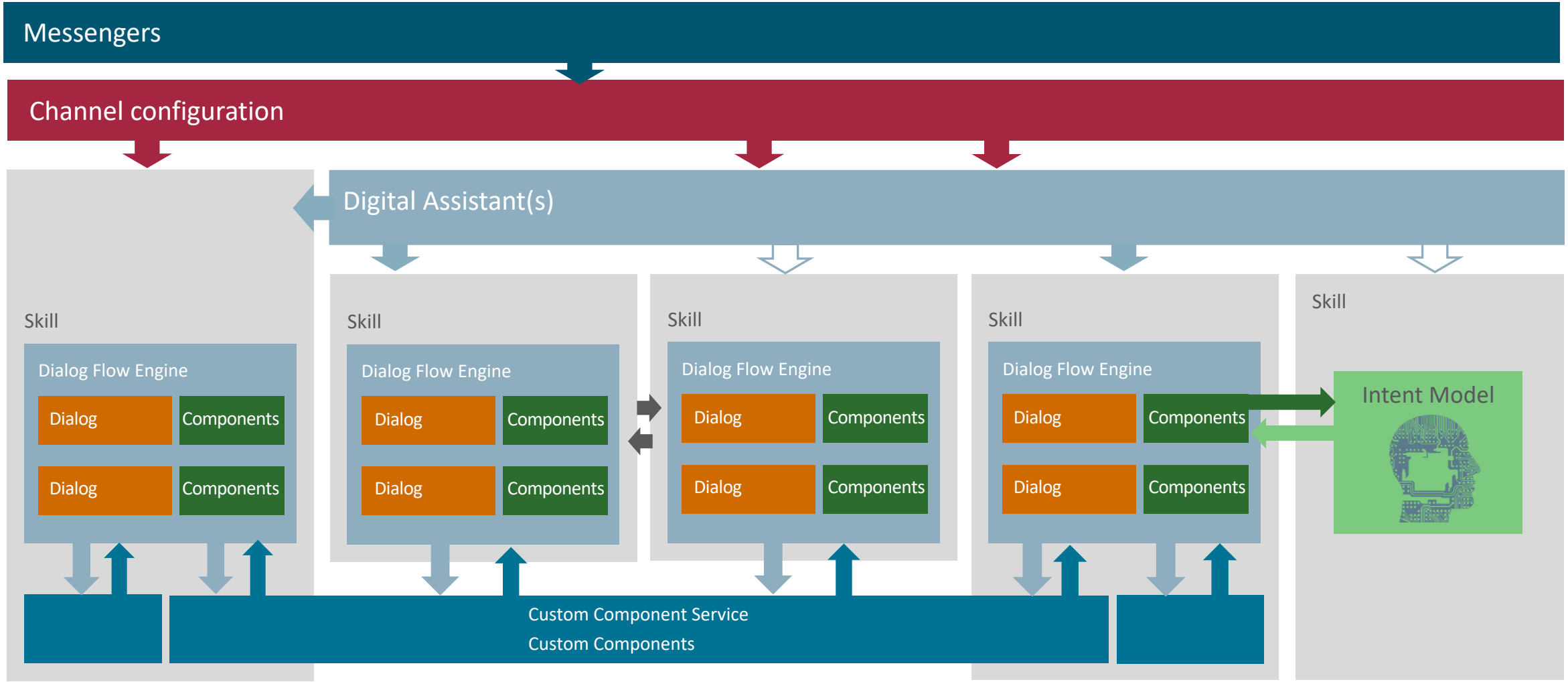


Image courtesy of pixabay.com

# Oracle Digital Assistant architecture possibilities

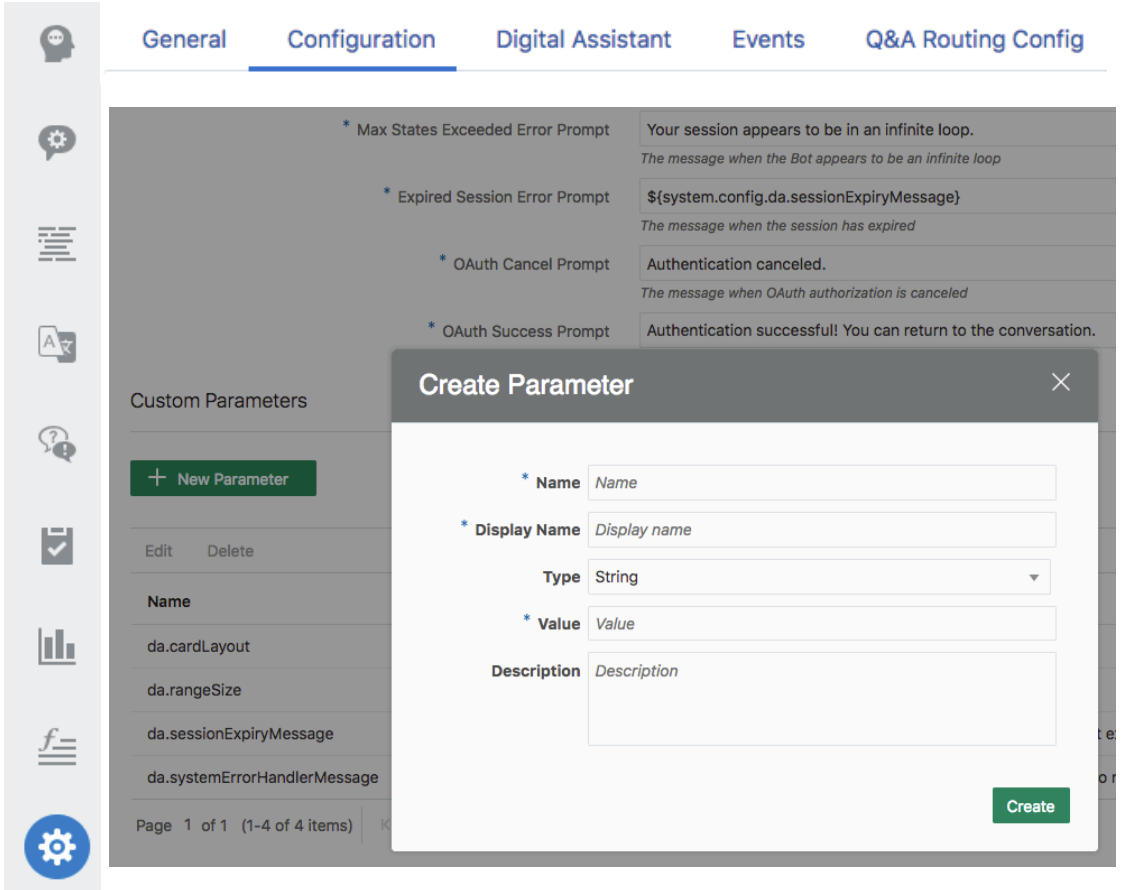


# Topic agenda

1. Architecture matters
2. Skill patterns
3. Digital assistant pattern
4. Skill Parameters

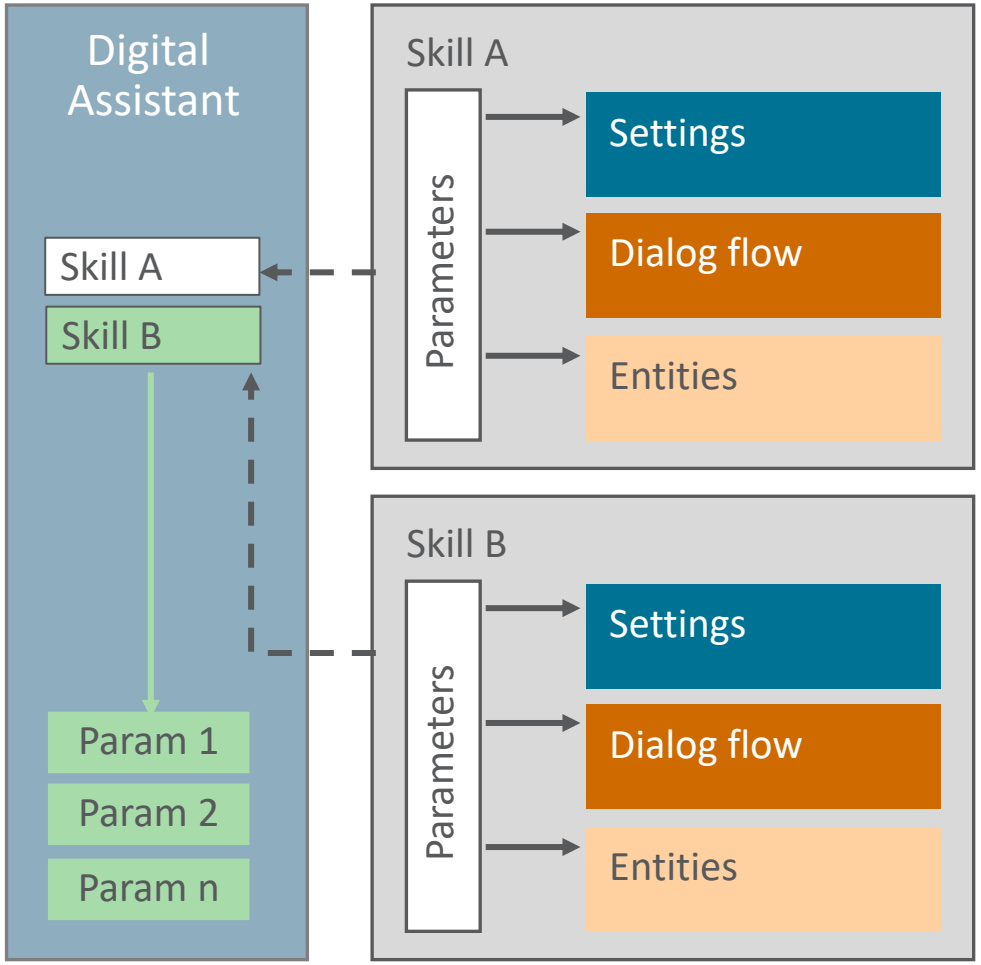
# Defining custom parameters for a skill

- Custom parameters are created in the *Settings* panel of a skill
- Supported data types are string, integer, float and boolean
- Naming convention
  - Parameter name **with no** "*da.*" prefix are in skill-only scope
  - Parameter names **with** "*da.*" prefix are visible and accessible in digital assistant



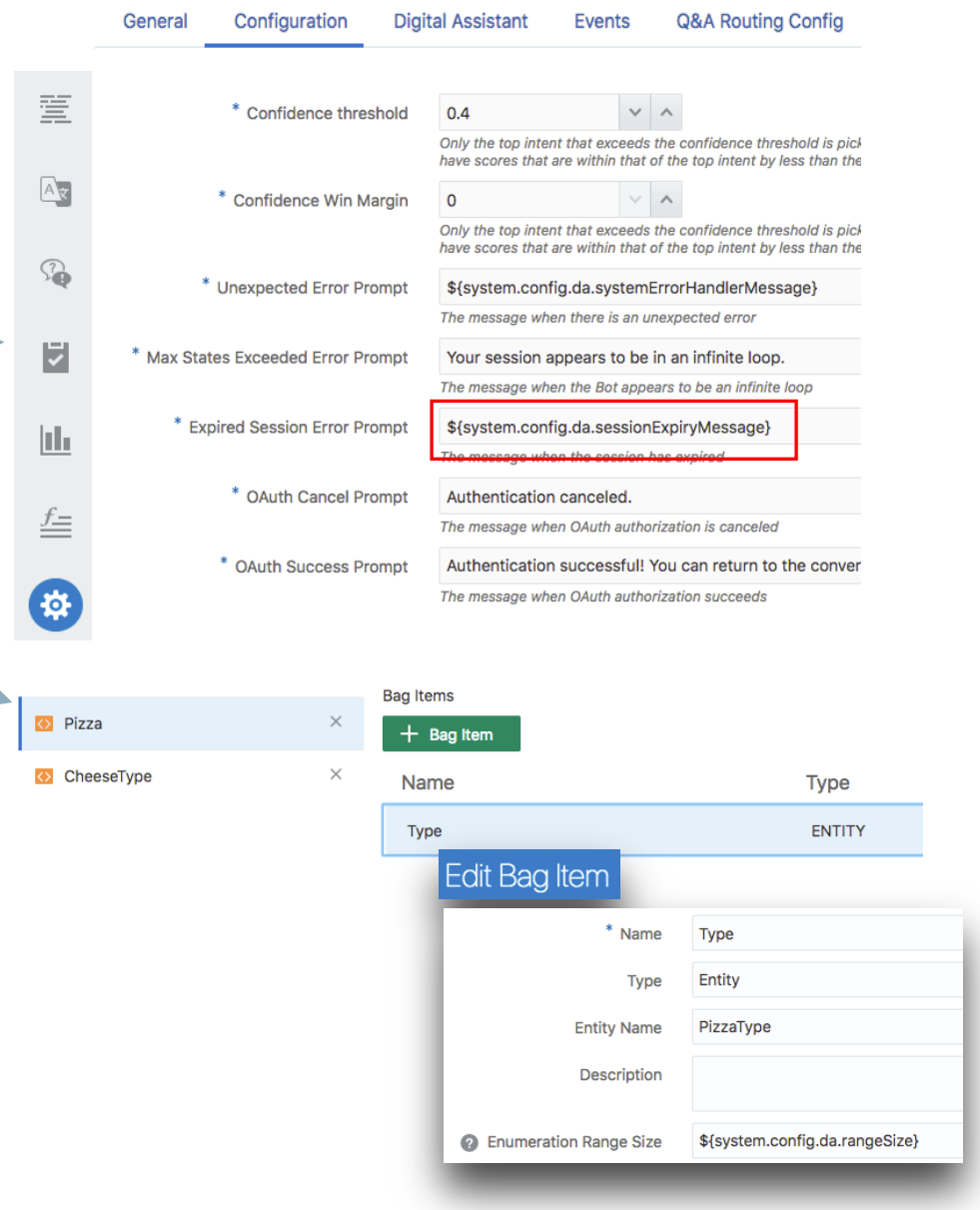
# Exposing skill parameters to digital assistant

- Improves user experience through consistency
  - Consistent messages
  - Similar behaviors and looks
- Defines a contract between a skill and the digital assistant
  - Improves reusability
  - Parameters with a "da." name prefix are exposed to digital assistant



# Accessing parameters in skill

- Skill parameters are accessible from
  - Skill's settings panel
  - In Entities
  - In dialog flow
- Read access
  - `${system.config.<name>}`
- Write access
  - `System.SetVariable`
    - variable: "system.config.<name>



# Integrated Cloud

## Applications & Platform Services


ORACLE®